

(19)日本国特許庁(J P)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平6-83598

(43)公開日 平成6年(1994)3月25日

(51)Int.Cl.<sup>5</sup>

G 0 6 F 9/06

識別記号

4 3 0 E 9367-5B

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数7(全 20 頁)

(21)出願番号 特願平4-235828

(22)出願日 平成4年(1992)9月3日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(71)出願人 391002409

日立システムエンジニアリング株式会社

東京都大田区大森北3丁目2番16号

(72)発明者 秋庭 真一

神奈川県川崎市幸区鹿島田890番地の12

株式会社日立製作所ビジネスシステム開発

センタ内

(74)代理人 弁理士 鈴木 誠

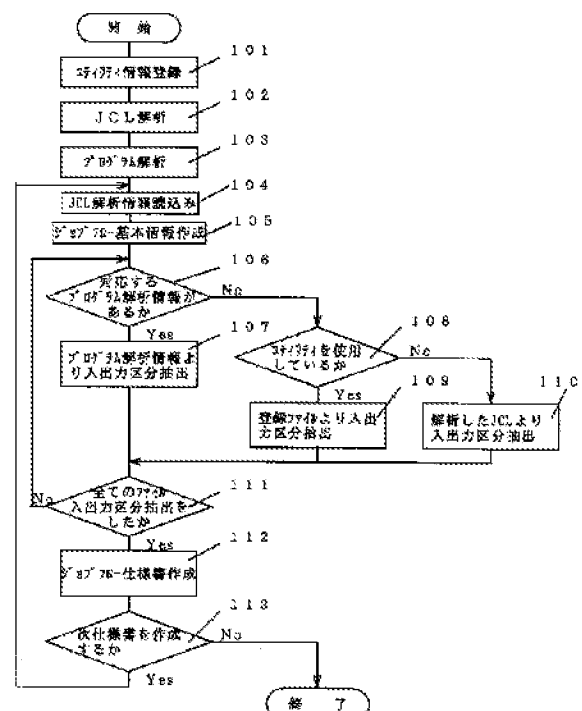
最終頁に続く

(54)【発明の名称】 ジョブフロー仕様書自動作成方法

(57)【要約】

【目的】ジョブ制御文やソースプログラム、データベース定義仕様等から、正確なジョブフロー仕様書を自動的に作成することを可能にする。

【構成】 プログラム解析が行えないユーティリティに関するファイル入出力区分を予め登録する(101)。ジョブ制御文の解析(102)、ソースプログラムの解析(103)を行い、仕様書自動作成の対象となるジョブ制御文を入力し(104)、ジョブフロー基本情報を作成する(105)。そして、入力したジョブ制御文中のロードモジュール名より、対応するソースプログラム解析情報、ユーティリティ情報またはジョブ制御文解析情報のいずれかよりファイル入出力区分を抽出し(106~111)、ジョブフロー仕様書を自動的に作成する(112)。



## 【特許請求の範囲】

【請求項 1】 ジョブ制御文とソースプログラムを解析してジョブ制御文中の各ロードモジュールの実行順序と使用するファイルの入出力区分を判定して、ジョブ内のプログラムの実行順序とファイルの入出力を含む図式表現のジョブフロー仕様書を計算機の支援により自動作成する作成方法において、

予め少なくともユティリティ名とその使用ファイル名、ファイル入出力区分を示すユティリティ情報を記憶し、ジョブ制御文の解析によって得られるロードモジュール名よりユティリティであるかを前記情報より識別し、ユティリティと識別したロードモジュールに対しては、該情報より使用ファイルのファイル入出力区分を決定してジョブフロー仕様書を作成することを特徴とするジョブフロー仕様書自動作成方法。

【請求項 2】 ジョブ制御文中のファイルの作成・削除・保持を示す情報の組合せ毎に、ファイル入出力区分を定めた情報を蓄積しておき、ロードモジュール名と同じ名称を持つソースプログラムがない場合は、該情報から当該ロードモジュールの使用ファイルの入出力区分を決定し、ジョブフロー仕様書を作成することを特徴とする請求項 1 記載のジョブフロー仕様書自動作成方法。

【請求項 3】 データベース定義仕様を解析して、該ロードモジュールが入出力するデータベース名と入出力区分情報を得て、データベースに関する情報をジョブフロー仕様書に出力することを特徴とする請求項 1 または 2 記載のジョブフロー仕様書自動作成方法。

【請求項 4】 ジョブ制御文中のロードモジュールに対するソースプログラム名を対話により指定し、ソースプログラムの解析情報から使用ファイルの入出力区分を取得することを特徴とする請求項 1、2 または 3 記載のジョブフロー仕様書自動作成方法。

【請求項 5】 ジョブ制御文の解析で抽出したロードモジュール名がユティリティであることを示す情報、対応するプログラム解析情報の有無を示す情報、データベースを使用の有無を示す情報の少なくともいづれかを予め作成してから、該情報を利用してジョブフロー仕様書を作成することを特徴とする請求項 1、2、3 または 4 記載のジョブフロー仕様書自動作成方法。

【請求項 6】 請求項 1、請求項 2、請求項 3、請求項 4 および請求項 5 記載のジョブフロー仕様書自動作成方法において、作成された複数のジョブフロー仕様書情報に基づき、ファイルとジョブまたはファイルとプログラムとの関連情報を作成して出力することを特徴とするジョブフロー仕様書自動作成方法。

【請求項 7】 請求項 1、請求項 2、請求項 3、請求項 4、請求項 5 および請求項 6 記載のジョブフロー仕様書自動作成方法において、作成されたジョブフロー仕様書に対し、必要に応じて該仕様書を編集し、編集された該仕様書から新しいジョブ制御文を自動作成することを特

徴とするジョブフロー仕様書自動作成方法。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 本発明は、ソフトウェア開発の計算機による支援システムに係り、特にソフトウェアの保守・理解、更に新規ソフトウェア開発作業に好適なジョブフロー仕様書の自動作成方法に関する。

## 【0002】

【従来の技術】 一般にソフトウェアの開発では、開発すべきソフトウェアの具備すべき機能を定め、その機能を実現するソフトウェアを段階的に詳細化して設計を進め、最終的にプログラムを作成する方法が用いられている。

【0003】 設計結果は、詳細化の各段階ごとに設計仕様書として記述され、次の設計段階では、前の段階で作成された仕様書の各要素を詳細化した仕様書が作成される場合が一般である。

【0004】 こういった方法では、新規にソフトウェアを開発する場合には有効であるが、これを用いて開発したソフトウェアの保守を行うには、次のような問題点がある。すなわち、初期の段階で作成された上位仕様書と、これを詳細化して作成されたソフトウェア生産物（プログラム、ジョブ制御文、下位仕様書）と内容が一致しない場合が生じる。この場合、仕様書は実際のプログラムやジョブ制御文の内容を表さなくなり、保守が困難になる。

【0005】 このような問題を解決するには、詳細化された仕様書、ソースプログラム、ジョブ制御文などの下位ソフトウェア生産物から上位仕様書を作成することにより、上位の仕様書が実際のソースプログラムやジョブ制御文の内容と一致させる必要がある。このための従来技術としては、例えば特願平 1-237726 号がある。ここでは、複数の詳細化された仕様書あるいはソースプログラムやジョブ制御文などのソフトウェア生産物を解析して、その中の上位仕様書に含むべき仕様情報を抽出し、この仕様情報を上位仕様書の形式に変換している。

## 【0006】

【発明が解決しようとする課題】 上記特願平 1-237726 号は、下位のソフトウェア生産物から上位のソフトウェア生産物を作成する基本的な方法を示しているが、ジョブ内のソースプログラムの実行順序や、ファイルの入出力を表すジョブフロー仕様書を作成する場合、以下のような問題点が残されている。

【0007】 (1) ジョブ内で実行されるソースプログラムがユティリティの場合は、対応するソースプログラムを解析することができないので、正確なジョブフロー仕様書を自動作成することができない。

【0008】 (2) ソースプログラムが紛失している場合なども、ソースプログラムを解析して、使用するファ

イルの入出力区分を判定させることができないので、正確なジョブフロー仕様書を自動作成することができない。

【0009】(3) データベースを使用している場合、ジョブ制御文やソースプログラムの解析だけではそのデータベースの入出力が判定できないので、正確なジョブフロー仕様書を自動作成することができない。

【0010】(4) ジョブ制御文中で記述されているロードモジュール名とソースプログラム名が一致しない場合、対応するソースプログラムを見つけることができないので、正確なジョブフロー仕様書を自動作成することができない。

【0011】(5) 大量の既存ソフトウェア資源から上位の仕様書を作成する場合、処理効率が悪い。

【0012】(6) 作成するのは上位仕様書のみであり、保守や理解支援、変更影響範囲の洗い出し作業を効率的に行えるようにするための十分な情報を出力することができない。

【0013】(7) 作成されたジョブフロー仕様書を編集して、新規にジョブ制御文を自動作成することができない。

【0014】本発明の第1の目的は、正確なジョブフロー仕様書を自動的に作成することを可能にすることにある。

【0015】本発明の第2の目的は、処理効率良くジョブフロー仕様書の自動作成を行えるようにすることである。

【0016】本発明の第3の目的は、保守や理解支援、変更影響範囲の洗い出し作業を効率的に行えるようにするための情報を出力することにある。

【0017】本発明の第4の目的は、作成されたジョブフロー仕様書の編集や、該編集された該仕様書から新しいジョブ制御文を自動的に作成することを可能にすることにある。

【0018】

【課題を解決するための手段】請求項1の発明は、ジョブ制御文やプログラムから上位仕様情報であるジョブフロー仕様書を作成する際に、ユティリティに関して少なくともユティリティ名、その使用ファイル名、ファイル入出力区分情報を持つ情報を予め蓄積しておき、ジョブ制御文中に記述されているロードモジュール名が該情報のユティリティ名と一致した場合には、該情報中のファイル入出力区分情報を用いて、ジョブフロー仕様書を自動作成するものである。

【0019】請求項2の発明は、ジョブ制御文中のファイルの作成・削除・保持を示す情報の組合せに対し、各組合せにおいてのファイル入出力区分を定めたテーブルを予め蓄積しておき、ロードモジュール名と同じ名称を持つソースプログラムやユティリティが無い場合は、該テーブルから使用ファイルの入出力区分を決定し、ジョ

ブフロー仕様書を自動作成するものである。

【0020】請求項3の発明は、予めデータベース定義仕様を解析しておき、ジョブ制御文またはソースプログラムとデータベース定義仕様の解析情報からデータベースを使用していると判断された時には、該解析情報より使用するデータベース名と入出力区分の情報を得て、ジョブフロー仕様書を自動作成するものである。

【0021】請求項4の発明は、ジョブ制御文の解析によって得られるロードモジュールに対応するプログラム名をディスプレイ画面上で対話形式で指定し、指定されたソースプログラムの解析情報から使用ファイルと入出力区分を取得し、ジョブフロー仕様書を自動作成するものである。

【0022】請求項5の発明は、ジョブ制御文、ソースプログラム、データベースの解析情報とユティリティに関する情報より、ジョブ制御文解析で抽出したロードモジュール名がユティリティであることを示す情報、対応するプログラム解析情報の有無やデータベースを使用の有無を示す情報を自動的に作成してから、該情報を利用してジョブフロー仕様書を自動作成するものである。

【0023】請求項6の発明は、上記のようにして作成された複数のジョブフロー仕様書情報に基づき、ファイルとそれを使用しているジョブの関連情報、またはファイルとそのファイルを使用しているプログラムの関連情報を作成して出力するものである。

【0024】請求項7の発明は、上記のようにして作成されたジョブフロー仕様書をディスプレイ画面上で対話形式で編集し、その編集されたジョブフロー仕様書に基づき新しいジョブ制御文を自動的に作成するものである。

【0025】

【作用】第1の目的に対しては、ジョブ制御文やプログラムから上位仕様情報であるジョブフロー仕様書を作成する際に、ユティリティに関して少なくともユティリティ名、その使用ファイル名、ファイル入出力区分情報を持つ情報を蓄積しておき、ジョブ制御文中に記述されているロードモジュール名が該情報のユティリティ名と一致した場合には、該情報中のファイル入出力区分を決定させることで、ユティリティを使用している場合においても、正確なジョブフロー仕様書が自動作成できる。

【0026】また、ジョブ制御文中のファイルの作成・削除・保持を示す情報の組合せに対し、各組合せにおいてのファイル入出力区分を定めた情報を蓄積しておき、ジョブ制御文中に記述されているロードモジュール名と同じ名称を持つソースプログラムが無いロードモジュールに対しては、該情報から使用ファイルの入出力区分を決定させることで、ロードモジュールに対する解析情報が無い場合においても、正確なジョブフロー仕様書が自動作成できる。

【0027】また、予めデータベース定義仕様を解析し

ておき、該解析情報より使用するデータベース名と入出力区分を得ることで、データベースを使用している場合においても、正確なジョブフロー仕様書が自動作成できる。

【0028】また、ロードモジュール名に対応するプログラム名をディスプレイ画面上で対話形式で指定することができるため、ロードモジュール名と名称が違っているためにソースプログラムと対応付けができなかった場合においても、指定されたソースプログラム解析情報から使用するファイルの入出力区分を決定し、自動的にジョブフロー仕様書を作成することができる。

【0029】第2の目的に対しては、ジョブ制御文、ソースプログラム、データベースの解析情報とユティリティに関する情報より、ジョブ制御文の解析で抽出したロードモジュール名がユティリティであるかを示す情報、対応するプログラム解析情報の有無を示す情報やデータベースの使用の有無を示す情報を予め作成し、該テーブルを利用してジョブフロー仕様書を作成することで、それぞれの情報に対して行う検索処理を必要最小限になり、ジョブフロー仕様書が効率良く自動作成される。

【0030】第3の目的に対しては、ジョブ制御文、ソースプログラム、データベースの解析情報とユティリティに関する情報より、ジョブ制御文解析で抽出したロードモジュール名がユティリティであるか、対応するプログラム解析情報の有無やデータベースを使用の有無を示す情報からファイルとそれを使用しているジョブの関連情報及びファイルとそのファイルを使用しているプログラムの関連情報を作成し、出力することで、あるソースプログラムまたはジョブ制御文に修正が行われた場合、他のどこに影響があるかを分析するため情報が出力され、保守や変更影響範囲の洗い出し作業を効率的に行うことができるようになる。

【0031】第4の目的に対しては、下位ソフトウェア生産物であるジョブ制御文、ソースプログラム、データベース定義仕様等より作成したジョブフロー仕様書をディスプレイ画面上で編集し、編集された該仕様書からジョブ制御文を自動作成させることを可能にするため、システムの再開発/再構築作業の効率化に役立つことができるようになる。

#### 【0032】

【実施例】図2は、本発明に係わるジョブフロー仕様書自動作成方法を達成するシステムの構成例の概略ブロック図を示す。図2において、1はプログラムに基づく逐次処理機能を有するCPUであり、ここでは、後述の図1、図11、図17あるいは図23の手順の実行を制御する。2はキーボードとマウスとディスプレイ画面を有する対話端末であり、該システムを用いる作業者が指示を入力したり、結果を該作業者に表示するために用いる。3は図1、図11、図17あるいは図23の手順を実現するためのプログラム、処理の中間結果などが格納

される主メモリである。4は該システムで用いられる既存ソフトウェア資源が格納されているファイルである。5は作業によって登録された情報が格納されているファイルである。6は該システムによって作成されたジョブフロー仕様書等の情報が格納されるファイルである。7はプリンタ装置であり、ここでは該システムによって作成されたジョブフロー仕様書等が出力される。以下、本発明の各実施例について詳述する。

【0033】＜実施例1＞図1は、本発明に係わるジョブフロー仕様書自動作成方法の第1の実施例の処理手順を示すフローチャートである。以下に、図1のフローチャートに基づき第1の実施例を説明する。

【0034】まず、解析対象のジョブ制御文（以下、JCLと称す）で使用しているユティリティに関する情報を予め登録する（ステップ101）。その登録情報の例を図3に示す。301はユティリティの名称で、そのユティリティが使用するDD名（302）と各々のDDのファイル入出力区分（303）を設定する。また、そのファイルが一時的であるときは、304に示すようなフラグを付ける。

【0035】次に、該ジョブフロー仕様書自動作成の対象となる全てのJCLの解析（ステップ102）及びソースプログラムの解析（ステップ103）を行う。

【0036】図4は解析対象のJCLの例であり、401はジョブ名、402はジョブステップ名、403はロードモジュール名、404は物理ファイル名、405はDD名、406はDISPオペランドのパラメータを示す。図4のJCLの解析結果は、図5に示したようなテーブルに格納される。例えばジョブ名（501）には401の「JOB1」、ジョブステップ名（502）には402の「STEP1」、ロードモジュール名（503）には403の「PROG01」、物理ファイル名（504）には404の「SAMPLE1」、DD名（505）には405の「DD1」をそれぞれ設定し、さらに、ファイルの入出力区分（506）には、図6のテーブルに示すDISPオペランドのパラメータの組合せによって定義された情報に従って設定する。図6で、601はDISPオペランドのパラメータの組合せによって設定するファイルの入出力区分を示し、602はそのファイルが一時的なものであるかを示すフラグである。該テーブルの情報も予め登録/編集が可能である。

【0037】図7は解析対象のプログラムの例であり、701はプログラム名、702は論理ファイル名、703はDD名、704はレコード名、705はファイルの入出力区分を示す。図7の解析結果は、図8に示したようなテーブルに格納される。例えば、プログラム名（801）には701の「PROG01」、ファイル名（802）には702の「INF1LE」、DD名（803）には703の「DD1」、レコード名（804）には704からの「IN-REC」、ファイルの入出力区

分(805)には、図7の705に該レコードのREADが示されているので入力で見なし「I」を設定する。なお、WRITEの場合は出力で見なし、ファイルの入出力区分には「O」が設定される。

【0038】以上の解析が終了後、ジョブフロー自動作成を行うJCL解析情報(図5)を読み込む(ステップ104)。次に、ジョブフロー仕様書作成のためのジョブフロー基本情報を作成する(ステップ105)。ジョブフロー基本情報の例を図9に示す。図9は図5に対応するもので、例えば、ジョブ名(901)には501の「JOB1」、ジョブステップ名(902)には502の「STEP1」を設定し、ロードモジュール名(903)には503の「PROG01」、物理ファイル名(904)には504の「SAMPLE1」、DD名(905)には505の「DD1」を設定する。

【0039】次に、図9のファイル入出力区分(906)を設定するために、ロードモジュール名(903)と一致するプログラム名を持つプログラム解析情報(図8)があるかチェックする(ステップ106)。もし一致するプログラム名(801)が存在する場合は、そのプログラム解析情報で示すファイルの入出力区分(805)を抽出し、図9のファイル入出力区分(906)に追加する(ステップ107)。もしロードモジュール名(903)と一致するプログラム名を持つプログラム解析情報が無かった場合は、そのロードモジュール名903と一致するユーティリティ名がステップ101で設定されたテーブル(図3)に登録されているかをチェックする(ステップ108)。一致するユーティリティ名(301)が存在する場合は、テーブル中の303で示すファイルの入出力区分を抽出し、図9のファイル入出力区分(906)に追加する(ステップ109)。この時、テーブル中の一時ファイル区分に一時ファイル判定フラグ「Y」(304)が指定されていた場合は、図9の一時ファイル区分(907)に「Y」を設定する。また、ステップ108で一致するユーティリティ名が存在しなかった場合は、JCL解析情報(図5)のファイル入出力区分(506)から抽出し、図9のファイル入出力区分(906)に追加する(ステップ110)。

【0040】ステップ106~110の処理について、入力したJCL解析情報に関して全てのファイル入出力区分を抽出するまで繰り返す(ステップ111)。そして、全てのファイル入出力区分を抽出すると、該ジョブフロー基本情報よりジョブフロー仕様書を作成する(ステップ112)。その後、次のジョブフロー仕様書を作成する場合はステップ104に戻り(ステップ113)、該ジョブフロー自動作成の対象となる全てのJCL解析情報に対する処理を繰り返し実行し、ジョブフロー仕様書を作成する。

【0041】図10に、作成したジョブフロー仕様書の例を示す。ここで、1001は一般ファイルを示し、1

002は一般プログラムを示す。また、1003はユーティリティを示し、1004は一時ファイル、1005は出力装置であるプリンタを示す。また、1006は図9の904の物理ファイル名「SAMPLE1」であり、1007は905の該ファイルを入力するときのDD名「DD1」を示し、1008は902のジョブステップ名「STEP1」、1009は903のロードモジュール名「PROG01」を示す。

【0042】本実施例によれば、従来JCLだけでプログラム解析情報が得られないユーティリティなどの不明瞭であったファイルの入出力区分を、予め一度定義しておくだけで、ファイルの入出力区分を明確にしたジョブフロー仕様書が随時自動作成されるようになる。このため、作成した仕様情報を基に現システムの保守・理解に役立つ情報を提供できるようになる。

【0043】<実施例2>図11は本発明に係るジョブフロー仕様書自動作成方法の第2の実施例の処理手順を示すフローチャートである。本実施例では、第1の実施例に加え、使用するファイルがデータベースか、一般ファイルかを区別して、自動作成されたジョブフロー仕様書においてデータベースの使用状況を視覚的に表現させることを可能にするものである。図11において、図1のステップ105が1104に、ステップ112が1105に変わり、1101、1102、1103のステップが追加された以外は図1と同じである。

【0044】まず、第1の実施例と同様に、ユーティリティ情報登録、JCLの解析ソースプログラムの解析まで行い、次にデータベース定義仕様の解析を行う(ステップ1101)。図12は、あるデータベース定義仕様の例であり、1201はアクセス定義仕様書名、1202はデータベース名、1203はアクセス権限情報、1204はプログラム名、1205は物理ファイル名を示す。この解析結果は、図13に示すようなテーブルに格納される。例えばアクセス仕様名称(1301)には1201の「RSVRD」、データベース名(1302)には1202の「SAMPLEDB」、物理ファイル名(1303)には1205の「EXAM1」、使用プログラム名(1304)には1204の「PROG02」が設定される。そして、入出力区分(1305)には、1203の「SHRUPD」により、読み込みも書き込みも可能と定義されているので、入力と出力「IO」と設定される。

【0045】次に、ジョブフロー自動作成を行うJCL解析情報を読み込み(ステップ104)、ジョブフロー基本情報の作成を行う(ステップ1104)。ジョブフロー基本情報は図9とほぼ同様であるが、図9に示される物理ファイル名(904)とDD名(905)は設定しない。そして、ステップ1102で、データベースを使用しているか否かを、JCL中のロードモジュール名がデータベースを使用する場合に用いられる特定名称で

あるか、またデータベース定義仕様で定義されているソースプログラム名とソースプログラム解析で得られるプログラム名が一致しているかで判断し、使用している場合には、ステップ1103で、ジョブフロー基本情報に図13の1302のデータベース名と1303の物理ファイル名と1304の入出力区分を設定する。一致していない場合は、実施例1と同様な処理手順で、ファイル入出力区分を抽出する。

【0046】図14は、JCL中でデータベースを使用する場合に用いられる特定名称がロードモジュールに記述されている場合の例を示しており、該データベース解析情報を取り込んで作成されたジョブフロー基本情報を図15に示す。図14の1401で囲まれているのが、該データベースを使用するJCL部である。該データベースを使用するJCL部について、図15のジョブフロー基本情報では、ジョブ名(1501)には1402の「JOB2」、ジョブステップ名(1502)には1403の「STEP2」、ロードモジュール名(1503)には1404の「PDMBTCH」、そして、データベース名(1504)には、図13のデータベース解析情報の1302の「SAMPLEDB」、物理ファイル名(1505)には1303の「EXAM1」、入出力区分(1506)には1305の「IO」が設定される。

【0047】該ジョブフロー基本情報よりジョブフロー仕様書を自動作成する(ステップ1105)。このステップ1105では、図1のジョブフロー仕様書作成(112)にデータベースを使用している場合に視覚的に明示させることを追加している。図16に該ジョブフロー仕様書を例示する。1601はデータベースを一般ファイルと区別をして表現したもので、1602は1504のデータベース名を示し、1603は1505のデータベースで使用している物理ファイル名を示し、1604は1506の入出力区分を示している。

【0048】本実施例によれば、図10に示した第1の実施例のジョブフロー仕様書に加え、データベースの使用情報を含むジョブフロー仕様書を自動作成することができる。

【0049】＜実施例3＞図17は、本発明に係るジョブフロー仕様書自動作成方法の第3の実施例の処理手順を示すフローチャートである。図11で示した第2の実施例では、ファイル入出力区分を抽出するためにプログラム解析情報があるか、登録したユティリティ情報があるか、またデータベースを使用しているのかといった各情報毎に検索処理を行っている。そのため、大量に既存ソフトウェア資源が存在する場合において検索に要する無駄な時間がかかり生じると考えられる。そこで本実施例では、第2の実施例よりも処理効率を向上させ、更により正確なジョブフロー仕様書を自動作成させるために、全解析が終了した後、各解析情報の関連のを持つテ

ーブルを自動的に作成する。ジョブフロー仕様書作成時には、該テーブルより関係付いた情報が格納されているファイルのみ検索してファイル入出力区分を抽出し、ジョブフローを自動的に作成できるように改良したものである。このため、図17では、図11に1701、1702、1703、1704、1706、1707が追加され、1102、106、108のステップを削除し、107、109、1103のステップを1705で示す1つのステップにまとめている。

【0050】まず、第2の実施例と同様に、データベース解析まで行い(ステップ101、102、103、1100)、解析終了後に各情報の関連を示す情報テーブルを自動的に作成する(ステップ1701)。図18に該テーブルを例示する。図18において、1801にはJCL解析によって得られるロードモジュール名(図5の503)、1802にはそれに対応するプログラム名(図8の801)もしくはユティリティ名(図3の301)が設定される。ロードモジュールとプログラムの対応は、プログラムが格納されているメンバ名もしくはプログラム中に記述されているプログラムIDによって対応付けが行われる。該テーブルを作成した後に、関連情報を編集する必要があるか判定し(ステップ1702)、編集の必要があれば、ディスプレイ画面上で対話形式でプログラム名(1802)とプログラム区分(1803)、使用しているデータベース名(1804)を編集する(ステップ1703)。

【0051】次に、ジョブフロー基本情報を作成し(ステップ1104)、抽出したロードモジュール名に関連する情報があるかを上記テーブルから判定し(ステップ1704)、対応する情報がある場合は、該テーブル内にある情報から該ロードモジュールに対応する情報が格納されているファイルのみを検索し、ファイル名とそのファイル入出力区分を抽出する(ステップ1705)。関連する情報が無い場合は、直接DISPオペランドの組合せによって定義されたテーブルからファイル名とそのファイル入出力区分を抽出する(ステップ110)。これを1ロードモジュール内のすべてのファイルに対してファイル名とそのファイル入出力区分を抽出する(ステップ1706)。

【0052】次に、各情報の関連(JCL/ソース、JCL/DB)を示す情報テーブルを使用した場合と使用しない場合の処理の違いについて説明する。図19は、第2の実施例で示された手順により、ジョブフロー自動作成する例を示し、図20は、本実施例により作成された該テーブルを使用してジョブフロー仕様書を自動作成する手順を示している。本例では、ロードモジュール名「TESTPROG」に対応する解析情報が全くない場合の例を示す。

【0053】図19では、ジョブフロー自動作成の対象となるJCL解析情報からロードモジュール名を抽出し

(1901)、次にデータベースを使用しているかデータベース解析情報を検索し(1902)、無いと判断されると、次に該ロードモジュール名と一致するプログラム名を持つプログラム解析情報が存在するか検索し(1903)、無いと判断されると、次にユティリティとして登録されているか検索を行う(1904)。本例のように、該ロードモジュールに対応する情報が無かった場合は、最後にJCL解析中にあるDISPオペランドに定義されたファイル入出力区分を抽出し(1905)、ジョブフロー仕様書を自動作成する(1906)。

【0054】図20では、まず、ジョブフロー自動作成の対象となるJCL解析情報からロードモジュール名を抽出(2001)し、本実施例で作成されたテーブル2000から関係する情報があるかを判定する(2002)。本例では、該ロードモジュールに対応する情報が存在しないので、直接DISPオペランドに定義されたファイル入出力区分を抽出し(2003)、ジョブフロー仕様書を自動作成する(2004)。以上のような手順の違いにより処理効率の向上が図れる。

【0055】また、図17のステップ1707において、先に作成された複数の該テーブル情報より図21のような物理ファイルとジョブの関連情報や、図22に示すような論理ファイルとプログラムの関連情報を作成して出力する。図21において、物理ファイル名(2101)にはJCL解析で得られる図5の504の「SAMPLE1」、同様にジョブ名(2102)には501の「JOB1」、ジョブステップ名(2103)には502の「STEP1」、DD名(2104)には505の「DD1」、ファイルの入出力区分(2105)には506の「I」が設定され、複数の該情報がプリンタ装置に出力される。該出力情報によって、どのジョブのどのジョブステップでどのファイルがアクセスされるか明確にできる。また、図22のように、論理ファイル名(2201)にはプログラム解析で得られる図8の802の「INF1LE」、同様にプログラム名(2202)には801の「PROG01」、DD名(2204)には803の「DD1」、ファイルの入出力区分(2205)には805の「I」、プログラム区分(2203)については図18の1803の「SOURCE」が設定され、複数の該情報がプリンタ装置に出力される。該出力情報によって、どのプログラムどのファイルがアクセスされるか明確にできる。これらの情報により、使用しているファイルに関する修正が行われる時などに影響する範囲を早期に発見することが可能となる。

【0056】本実施例によれば、第2の実施例よりも処理を効率良く行うことができる。また、JCL内のロードモジュール名とソースプログラム名が一致しない情報に関しても、ロードモジュール名と対応するプログラム名をディスプレイ画面上で対話形式で指定することができるため、指定したプログラムのファイル入出力区分を

自動的に取り入れた、より正確なジョブフロー仕様書を自動作成することができる。

【0057】<実施例4>図23は、本発明に係るジョブフロー仕様書自動作成方法の第4の実施例の処理手順を示すフローチャートである。本実施例は、図17に2301、2302、2303のステップを加えたものである。本実施例では、第1、第2、第3の実施例で自動作成されたジョブフロー仕様書に対して編集を行い、編集されたジョブフロー仕様書より新しいJCLを自動的に作成できるように改良したものである。

【0058】図23において、自動作成されたジョブフロー仕様書を基に該仕様書を編集するか判定し(ステップ2301)、編集する必要があると判断された場合は、ディスプレイ画面上でマウスやキーボードを用いて編集を行う(ステップ2302)。自動作成された編集前のジョブフロー仕様書を図10とし、該仕様書に編集を加えたジョブフロー仕様書の例を図24に示す。この編集では、図10で示されるジョブ名「JOB1」(901)のJCLに対して、ジョブステップ「STEP1」(902)のロードモジュール名「PROG01」(903)に物理ファイル名「TRAN1」(2402)、DD名「DD5」(2403)のファイルを入力する情報を自動作成されたジョブフロー仕様書に追加している。

【0059】次に、編集されたジョブフロー仕様書からJCLを自動的に作成する(ステップ2303)。自動的に作成されたJCLの例を図25に例示する。2501の「JOB1」と2502の「STEP1」と2503の「PROG01」は、自動作成したジョブフロー仕様書が既に持っていたJCL解析情報(図5)であり、本例で編集されて自動的に作成されたJCLは、2505のDD名に図24の2403の「DD5」、2504の物理ファイル名に2402の「TRAN1」、ファイル入出力区分は入力であるため、「DISP=SHR」(2506)が図10のジョブフロー仕様書を作成したJCLに追加される。

【0060】本実施例によれば、JCLやプログラム等から自動作成したジョブフロー仕様書を編集し、編集した該仕様書から新たにJCLを自動作成させることができる。

【0061】

【発明の効果】

(1) 請求項1の発明によれば、ジョブ内のユティリティの使用ファイルの入出力が判定できるので、正確なジョブフロー仕様書が自動作成される。

【0062】(2) 請求項2の発明によれば、対応するソースプログラムのないロードモジュールに対して、使用ファイルの入出力が判定できるので、正確なジョブフロー仕様書が自動作成される。

【0063】(3) 請求項3の発明によれば、使用する

10

20

30

40

50

データベースに対して、入出力が判断できるので、正確なジョブフロー仕様書が自動作成される。

【0064】(4) 請求項4の発明によれば、ロードモジュールに対し、対応するソースプログラム名を指定し、指定されたソースプログラムの解析情報から使用するファイルの入出力区分が判定できるため、正確なジョブフロー仕様書が自動作成される。

【0065】(5) 請求項5の発明によれば、予めロードモジュールとソースプログラム、データベース、ユーティリティの関連情報テーブルを作成してから、ジョブフロー仕様書を作成するので、ジョブフロー仕様書自動作成の処理効率を上げることができる。

【0066】(6) 請求項6の発明によれば、ファイル/ジョブ関連情報、ファイル/プログラム関連情報を作成し、出力するので保守や理解支援、変更影響範囲の洗い出し作業を効率的に行うことができる。

【0067】(7) 請求項7の発明によれば、作成されたジョブフロー仕様書の修正や修正されたジョブフロー仕様書から新しいJCLを自動作成できるので、システムの再開発/再構築の効率化に役立つことができる。

【図面の簡単な説明】

【図1】本発明の第1の実施例の処理手順を示すフローチャートである。

【図2】本発明のジョブフロー仕様書自動作成方法を達成するシステムの構成例を示すブロック図である。

【図3】第1の実施例で使用するユーティリティ登録情報の例である。

【図4】第1の実施例で使用するJCLの例である。

【図5】第1の実施例で使用するJCL解析情報の例である。

【図6】第1の実施例で使用するDISPオペランドの組合せ毎によるファイル入出力区分を決定させる定義情報の例である。

【図7】第1の実施例で使用するプログラムの例である。

【図8】第1の実施例で使用するプログラム解析情報の例である。

【図9】第1の実施例で作成されるジョブフロー基本情報の例である。

【図10】第1の実施例で作成されたジョブフロー仕様書の例である。

【図11】本発明の第2の実施例の処理手順を示すフローチャートである。

【図12】第2の実施例で使用するデータベース仕様定義の例である。

【図13】第2の実施例で使用するデータベース仕様定義の解析情報の例である。

【図14】第2の実施例で使用するJCLの例である。

【図15】第2の実施例で作成されるジョブフロー基本情報の例である。

【図16】第2の実施例で作成されるジョブフロー仕様書の例である。

【図17】本発明の第3の実施例の処理手順を示すフローチャートである。

【図18】第3の実施例で作成する関連情報テーブルの例である。

【図19】第3の実施例における使用ファイルとその入出力区分抽出手順の例を示す図である。

【図20】第3の実施例における使用ファイルとその入出力区分抽出手順の例を示す図である。

【図21】ファイル/ジョブ関連情報の例である。

【図22】ファイル/プログラム関連情報の例である。

【図23】本発明の第4の実施例の処理手順を示すフローチャートである。

【図24】図10のジョブフロー仕様書の編集後のジョブフロー仕様書の例である。

【図25】図24のジョブフロー仕様書より自動作成されたJCLの例である。

【符号の説明】

1 CPU

2 対話端末

3 主メモリ

4 既存ソフトウェア資源格納ファイル

5 ユーザ登録情報ファイル

6 ジョブフロー基本情報ファイル

7 プリンタ装置

101 ユティリティ情報登録ステップ

102 JCLの解析ステップ

103 ソースプログラムの解析ステップ

104 JCL解析情報入力ステップ

105, 1104 ジョブフロー基本情報作成ステップ

107, 109, 110, 1103, 1705 ファイル入出力区分抽出ステップ

112, 1105 ジョブフロー仕様書作成ステップ

1703 JCL/ソースプログラム、JCL/データベース関連情報作成ステップ

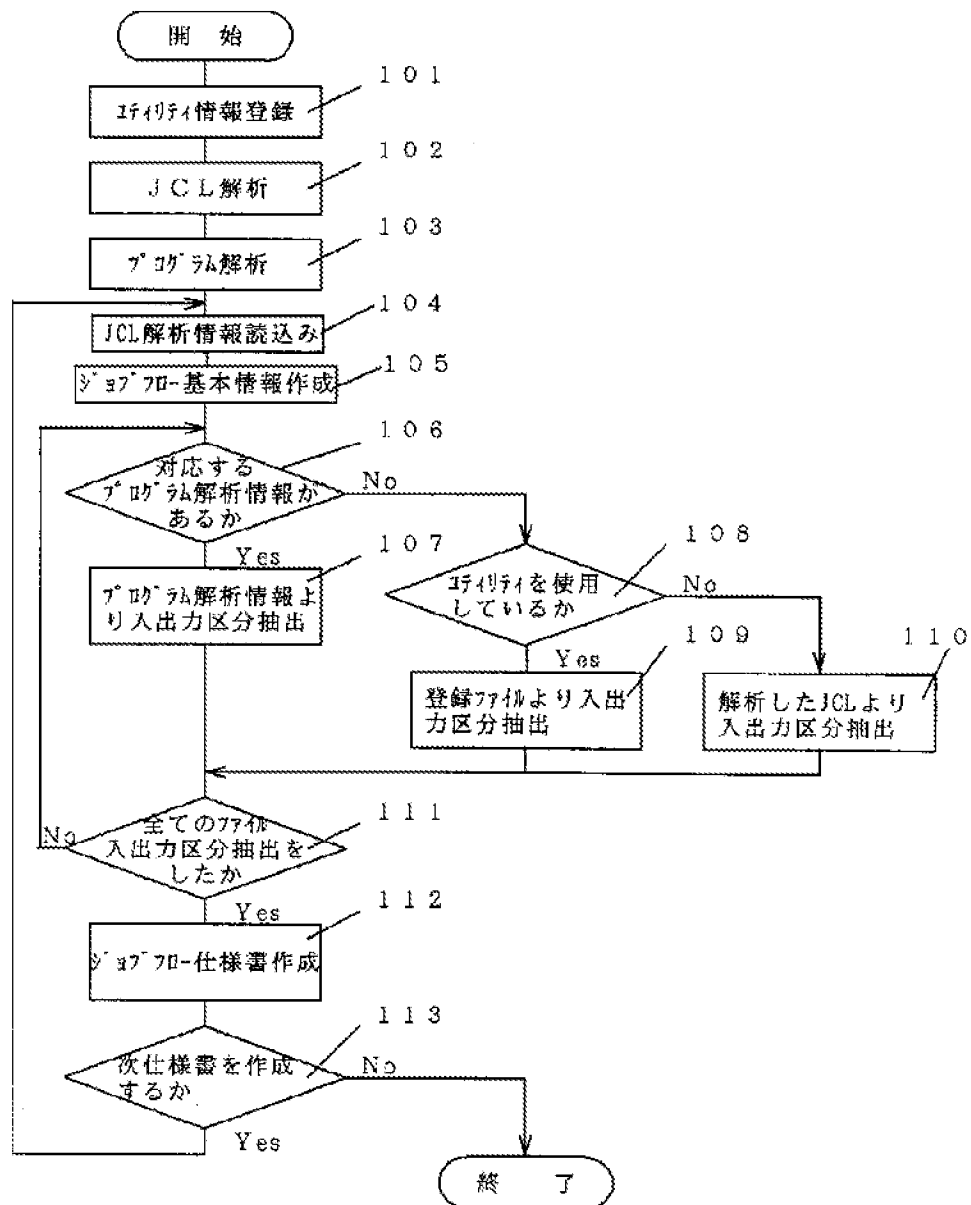
1707 ファイル/ジョブ、ファイル/プログラム情報作成ステップ

2302 ジョブフロー仕様書の編集ステップ

2303 JCL作成ステップ



【図1】

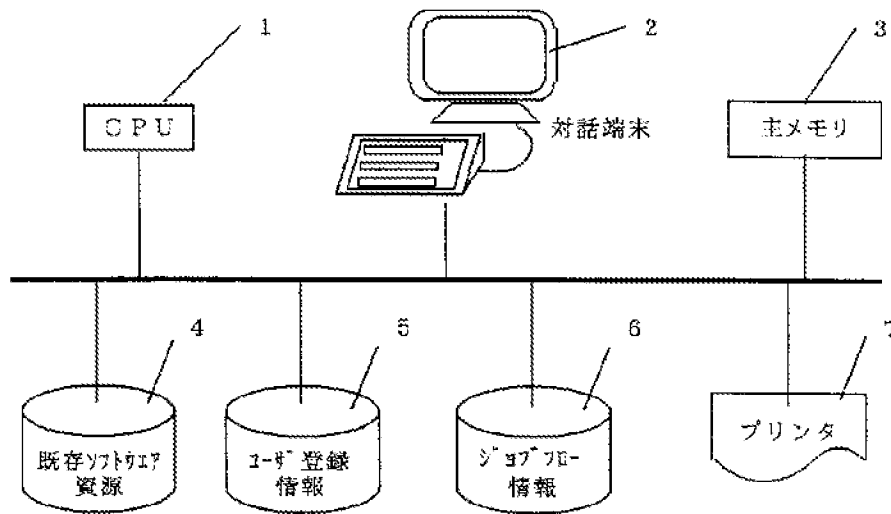


【図3】

ユーティリティ名	DD名	I/O区分	一時ファイル区分
SORT	SYSPRINT	0	Y
	SORTWK01	10	
	SORTIN	1	
	SORTOUT	0	

Figure 3 shows a table with 4 columns: ユーティリティ名 (Utility Name), DD名 (DD Name), I/O区分 (I/O Distinction), and 一時ファイル区分 (Temporary File Distinction). The table lists data for the SORT utility across four DDs: SYSPRINT, SORTWK01, SORTIN, and SORTOUT. The I/O distinctions are 0, 10, 1, and 0 respectively. The temporary file distinction is Y for all entries.

【図2】



【図4】

```

401 402 403 404 406
//JOB1 JOB
//STEP1 EXEC PGM=PROG01
405 //DD1 DD DSN=SAMPLE1, DISP=SHR
//DD2 DD DSN=SAMPLE2, DISP=SHR
//STEP2 EXEC PGM=TESTPROG
//DD3 DD DSN=SAMPLE2, DISP=SHR
//DD4 DD DSN=SortFIN, DISP=OLD
//STEP3 EXEC PGM=Sort
//SYSPRINT DD SYSOUT=A
//SORTWK01 DD UNIT=DISK, SPACE=(CYL,(5,1))
//SORTIN DD DSN=SortFIN, DISP=SHR
//SORTOUT DD DSN=SortFOUT, DISP=(NEW,KEEP,DELETE),
// UNIT=DISK, VOL=SER=VOL001, SPACE=(CYL,3),
// DCB=(RECFM=FB, LRECL=100, BLKSIZE=1000)
//SYSIN DD *
SORT FIELDS=(25,8,ZD,A), SIZE=E5000
/*
//

```

【図8】

プログラム名	論理ファイル名	DD名	レポート名	I/O区分
PROG01	INFILE	DD1	IN-REC	I
PROG01	OUTFILE	DD2	OUT-REC	O

801 802 803 804 805

【図5】

ジョブ名	ジョブステップ名	データ モジュール名	物理ファイル名	DD名	I/O区分	一時 ファイル区分
JOB1	STEP1	PROG01	SAMPLE1	DD1	I	
JOB1	STEP1	PROG01	SAMPLE2	DD2	I	
JOB1	STEP2	TESTPROG	SAMPLE2	DD3	I	
JOB1	STEP2	TESTPROG	SORTFIN	DD4	O	
JOB1	STEP3	SORT		SYSPRINT	O	
JOB1	STEP3	SORT		SORTWK01	IO	
JOB1	STEP3	SORT	SORTFIN	SORTIN	I	
JOB1	STEP3	SORT	SORTFOUT	SORTOUT	O	

【図6】

第一パラメタ	第二パラメタ	DELETE	KEEP	PASS	CATLG	UNCATLG	なし
NEW		I O	Y O	O	O	O	DELETEと同じ
MOD		I O	O	O	O	O	KEEPと同じ
RNW		I O	Y O	O	O	O	DELETEと同じ
OLD		I O	O	O	O	O	KEEPと同じ
SHR		I	I	I	I	I	KEEPと同じ
なし		NEWと同じ					

601 602

【図12】

```

1201 1202 1203
ASB=RSYRD DBM=SAMPLEDB ACCESS=SHRUPD
PROG=PROG02
.
.
DSN=EXAM1
LIST=EXAM1
.
.
DSN=EXAM2
LIST=EXAM2
END
1204
1205

```

【図7】

IDENTIFICATION PROGRAM-ID.	DIVISION. PROG01	701
CONFIGURATION	SECTION.	
INPUT-OUTPUT FILE-CONTROL.	SECTION.	702
SELECT INFILE ASSIGN TO DD1-DA-DK-S.		703
SELECT OUTFILE ASSIGN TO DD2-DA-DK-S.		
DATA FILE	DIVISION. SECTION.	
FD INFILE 01 IN-REC		704
FD OUTFILE 01 OUT-REC		
PROCEDURE	DIVISION.	
READ IN-REC		705
WRITE OUT-REC		

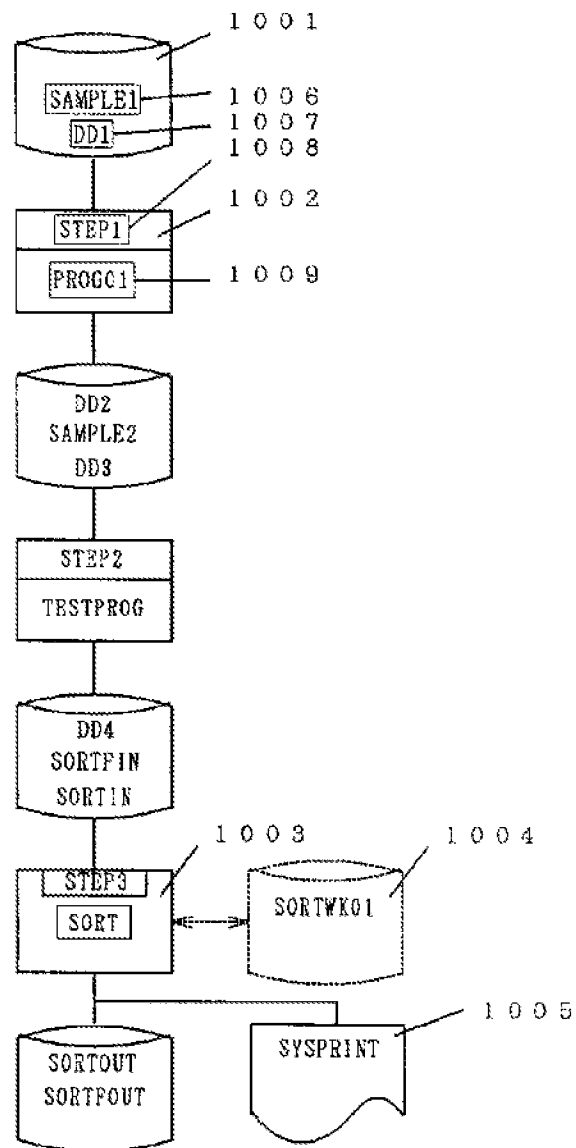
【図9】

ジョブ名	ジョブステップ名	ポート モジュール名	物理ファイル名	DD名	I/O区分	一時 ファイル区分
JOB1	STEP1	PROG01	SAMPLE1	DD1	I	
JOB1	STEP1	PROG01	SAMPLE2	DD2	0	
JOB1	STEP2	TESTPROG	SAMPLE2	DD3	1	
JOB1	STEP2	TESTPROG	SORTFIN	DD4	0	
JOB1	STEP3	SORT		SYSPRINT	0	
JOB1	STEP3	SORT		SORTWK01	10	Y
JOB1	STEP3	SORT	SORTFIN	SORTIN	1	
JOB1	STEP3	SORT	SORTFOUT	SORTOUT	0	

【図22】

論理ファイル名	プログラム名	プログラム 区分	DD名	I/O区分
INFILE	PROG1	SOURCE	DD1	I
OUTFILE	PROG9	SOURCE	DD9	0

【図10】



【図13】

物理定義 仕様書名	データベース名	物理ファイル名	使用プログラム名	I/O区分
RSVRD	SAMPLEDB	EXAM1	PROG02	IO
RSVRD	SAMPLEDB	EXAM2	PROG02	IO

1301

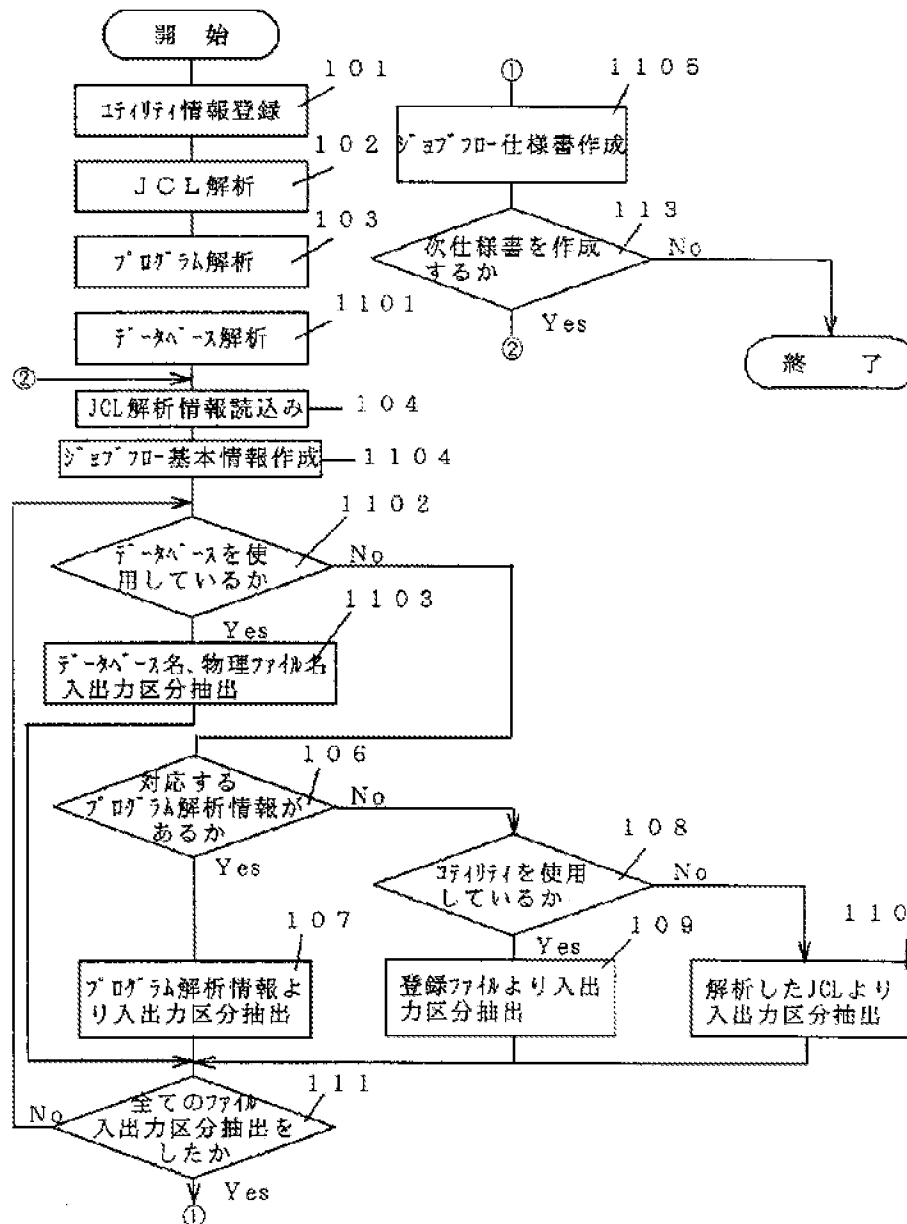
1302

1303

1304

1305

【図11】



【図14】

```

1402
//JOB2 JOB
//STEP1 EXEC PGM=PROG01
//DD1 DD DSN=SAMPLE1,DISP=SHR
//DD2 DD DSN=SAMPLE2,DISP=SHR
1403 //STEP2 EXEC PGM=PDMBTCH
//DD3 DD DSN=SAMPLE3,DISP=SHR
//PDMLIBD DD DSN=PDM2.LIBD,DISP=SHR
//PDMLIBM DD DSN=PDM2.LIBM,DISP=SHR
//EXAM1 DD DSN=PDM2.EXAM1,DISP=SHR
//EXAM2 DD DSN=PDM2.EXAM2,DISP=SHR
//PDM@IPT DD *
PDMBCH PGM=PROG02
1404
1401
1405
/*
//

```

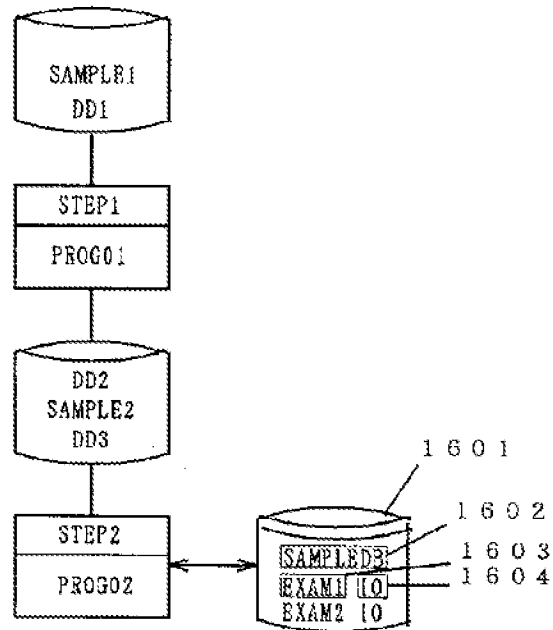
【図15】

ジョブ名	ジョブステップ名	ロードモジュール名	データヘッダ名	物理ファイル名	DD名	I/O区分	一時ファイル区分
JOB2	STEP1	PROG01		SAMPLE1	DD1	I	
JOB2	STEP1	PROG01		SAMPLE2	DD2	O	
JOB2	STEP2	PDMBTCH		SAMPLE2	DD3	I	
JOB2	STEP2	PDMBTCH	SAMPLEDB	EXAM1		IO	
JOB2	STEP2	PDMBTCH	SAMPLEDB	EXAM2		IO	
1501	1502	1503	1504	1505		1506	

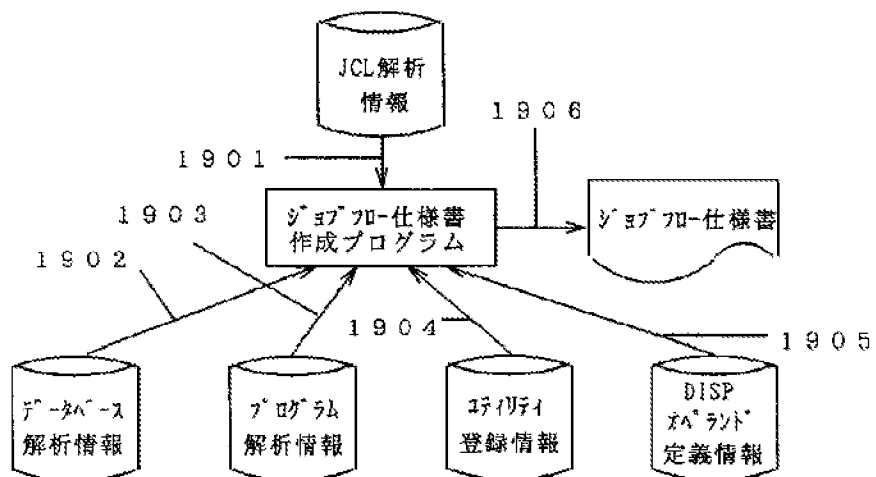
【図18】

ロードモジュール名	プログラム名	プログラム区分	データヘッダ名
PROG01	PROG01	SOURCE	
PROG02	PROG02	SOURCE	SAMPLEDB
SORT	SORT	UTILITY	
TESTPROG			
1801	1802	1803	1804

【図16】

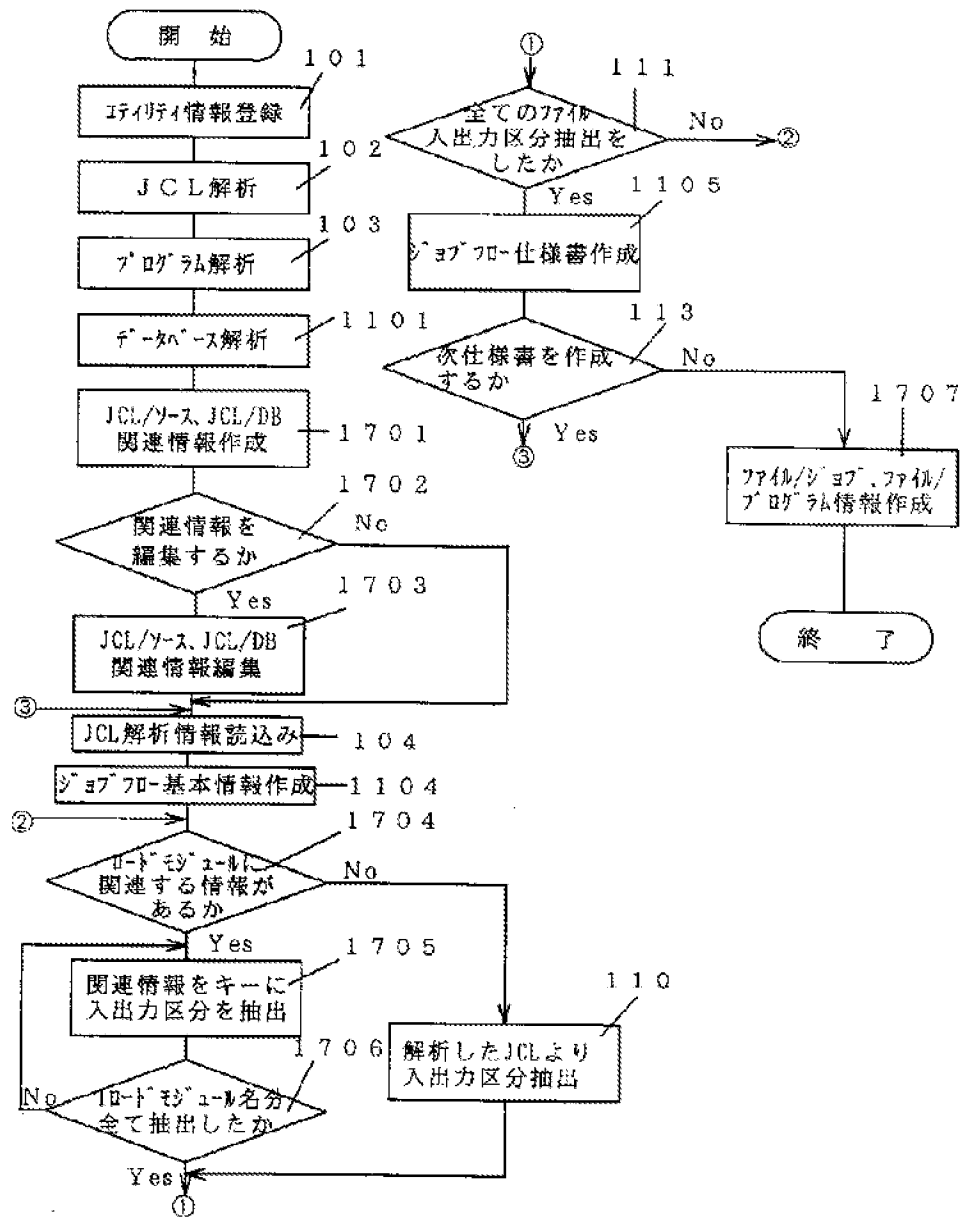


【図19】

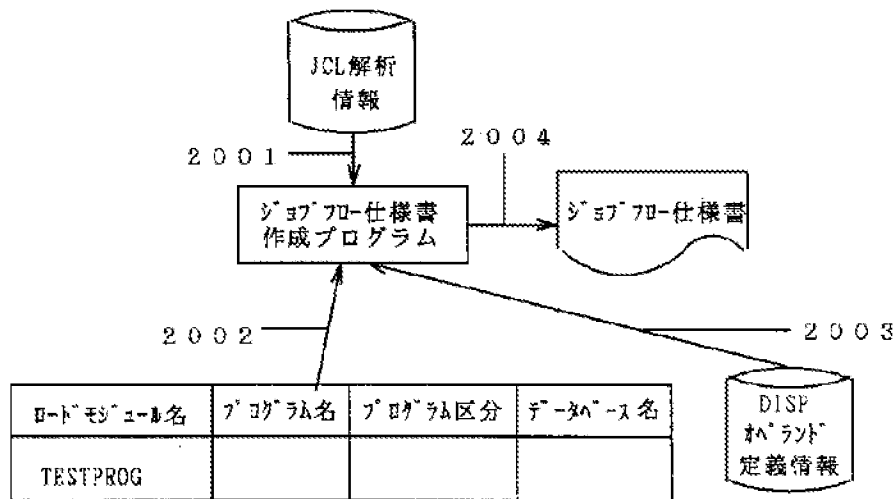




【図17】



【图 20】



【图 2-1】

物理ファイル名	ジョブ名	ジョブステップ名	ロードモジュール名	DD名	I/O区分	一時ファイル区分	データベース名
SAMPLE1 SAMPLE1	JOB1 JOB9	STEP1 STEP9	PROG01 PROG99	DD1 DD9	1 0		

2101      2102      2103      2104      2105      2106

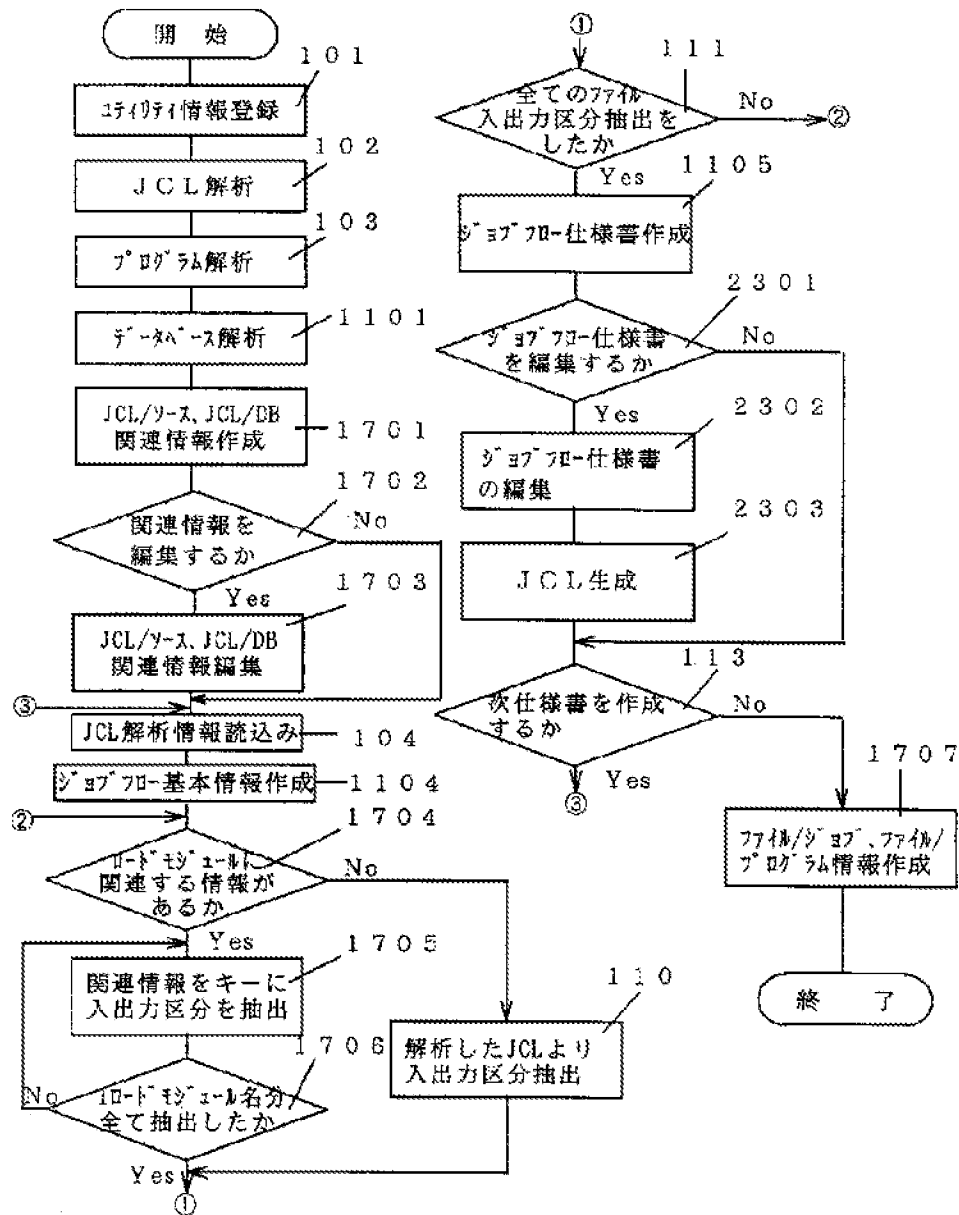
【图 2-5】

```

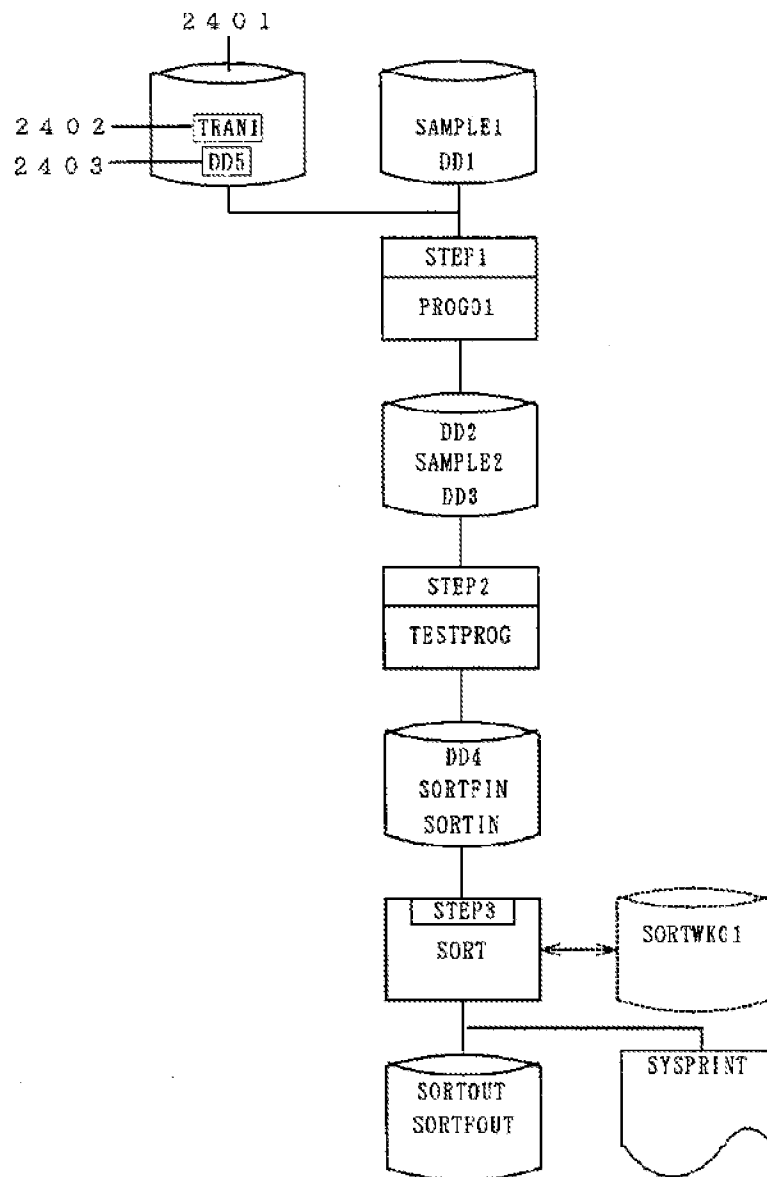
2501  //JOB1 JOB
2502  //STEP1 EXEC PGM=PROG01 2503
//DD1 DD DSN=SAMPLE1, DISP=SHR
//DD2 DD DSN=SAMPLE2, DISP=SHR
2505  //DD5 DD DSN=TRAN1, DISP=SHR 2504
//STEP2 EXEC PGM=TESTPROG
//DD3 DD DSN=SAMPLE2, DISP=SHR 2506
//DD4 DD DSN=SORTFIN, DISP=OLD
//STEP3 EXEC PGM= SORT
//SYSFRINT DD SYSOUT=A
//SORTWK01 DD UNIT=DISK, SPACE=(CYL, (5, 1))
//SORTIN DD DSN= SORTFIN, DISP=SHR
//SORTOUT DD DSN= SORTFOUT, DISP=(NEW, KEEP, DELETE),
// UNIT=DISK, VOL=SER=VOL001, SPACE=(CYL, 3),
// DCB=(RECFM=FB, LRECL=100, BLKSIZE=1000)
//SYSIN DD *
SORT FIELDS=(25, 8, ZD, A), SIZE=E5000
/*
//

```

【図23】



【図24】



フロントページの続き

(72)発明者 内藤 一郎  
神奈川県川崎市麻生区王禅寺1099番地 株  
式会社日立製作所システム開発研究所内

(72)発明者 月野 浩  
東京都大田区大森北三丁目2番16号 日立  
システムエンジニアリング株式会社内